Areo-LLM: A Distributed Framework for Secure UAV Communication and Intelligent Decision-Making

Balakrishnan Dharmalingam, Brett Piggott, Guohuan Feng, Rajdeep Mukherjee, Anyi Liu
Department of Computer Science and Engineering
Oakland University
Rochester, Michigan, U.S.A.
{bdharmalingam, bapiggott, gfeng, rajdeepmukherje, anyiliu}@oakland.edu

***Abstract*—In the context of increasing unmanned aerial vehicle (UAV) utilization in critical operations, ensuring secure and reliable communication between UAVs and Ground Control Stations (GCS) is crucial. This paper introduces Areo-LLM, a pioneering framework integrating various Large Language Models (LLMs) to enhance UAV mission security and efficiency. Unlike conventional LLMs that operate as large, singular entities, Areo-LLM employs a collaborative approach, leveraging smaller, specialized LLMs for tasks like inferencing, anomaly detection, and forecasting. These LLMs are strategically deployed across different computing environments, from onboard systems to cloud servers, facilitating a dynamic, distributed architecture that optimizes both performance and security. Areo-LLM's design focuses on task-specific fine-tuning through supervised techniques and reinforcement learning from human feedback, allowing precise adjustments to meet unique operational needs. The evaluation of Areo-LLM involved testing its integration and performance across various scenarios, demonstrating superior task-specific metrics and robust defense mechanisms against cyber threats. The results confirm that Areo-LLM not only enhances the decision-making and operational capabilities of UAV systems but also significantly lowers the risks of cyber exploits, setting a new standard for secure, intelligent UAV operations.**

***Index Terms*—unmanned aerial vehicle, large language models, anomaly detection and forecasting, edge computing**

I. Introduction

With the increasing deployment of unmanned aerial vehicle (UAVs) in mission-critical operations, securing the communication channels between UAVs and Ground Control Stations (GCS) becomes paramount. The integrity and confidentiality of the transmitted data must be ensured. In the evolving landscape of artificial intelligence, integrating Large Language Models (LLMs) with system and software security transforms defense mechanisms against cyber threats. LLMs, with their advanced capabilities in language comprehension and generation, offer significant potential to enhance detection and defensive strategies against cyber adversaries [1]–[6]. The data exchange between UAVs and GCS, particularly sensor data, involves time series with varying sampling rates and occasional data gaps, necessitating meticulous preprocessing to ensure the data's usability for fine-tuning LLMs.

In this paper, we present Areo-LLM, a novel framework that integrates different types of LLMs as a team for collaboration and information sharing for UAV flying missions. Specifically, different types of LLMs show capabilities in inferencing, anomaly detection, and forecasting. The LLMs team is strategically placed onboard, at the edge, or in the cloud to provide a robust and effective defense against various cyber exploits. Compared with the state-of-the-art (SOTA) foundation LLMs, such as Llama [7], Gemini [8], Mistral [9], and DBRX [10], Areo-LLM is not designed to outperform their benchmark capabilities. Instead, it fine-tuned special-skilled LLMs, which focus on particular tasks and data. The striking feature of Areo-LLM is two-fold. First, it leverages a team of LLMs, whose size might be considerably smaller than the versatile but all-in-one LLMs. Each LLM takes the cross-sectional or time-series data and accomplishes specific tasks accordingly. The distributed architecture also potentially introduces intelligent agents into the scene, which orchestrates and moderates the coordination among LLMs. Second, it allows the partial LLMs to be offloaded to the computing units on edge servers or cloud servers. Thus, some inference tasks can be performed by more powerful GPUs off-board. To train special-skilled LLMs, we collectively applied two fine-tuning technologies: 1) *supervised fine-tuning* (SFT); and 2) *reinforcement learning from human feedback* (RLHF). To evaluate the performance of Areo-LLM, we tested two types of LLM on different tasks. Specifically, we use small-scale LLMs (OPT-350m and OPT-125m) to mimic the UAV in network communication with the ground control station (GCS). We use time-sensitive LLMs (e.g., TimesNet) to detect anomalies and forecast future activities. Our evaluation results demonstrate that Areo-LLM is able to integrate the generative capabilities of various LLMs with high accuracy, in terms of accuracy, precision, recall, F1 score, and low error rates. Moreover, the fine-turned LLMs demonstrate low memory footprints in terms of VRAM usage. The contributions of this paper are summarized as follows:

- We designed and implemented Areo-LLM, a novel framework by assembling diverse LLMs for collaborative functioning, optimizing UAV missions through task specialization. This method advances beyond traditional large-scale LLMs by emphasizing precise, mission-critical ca-

pabilities enhanced through supervised fine-tuning and reinforcement learning from human feedback.
- The Areo-LLM promotes a scalable, distributed architecture that efficiently utilizes on-board, edge, and cloud computing resources, which ensures that Areo-LLM maintains high performance and cybersecurity standards across varied operational environments and computational capacities.
- We tested Areo-LLM's capabilities and observed superior task-specific performance, such as high accuracy, precision, recall, and F1 scores with minimal error rates. Moreover, its distributed nature offers robust protection against cyber threats, highlighting an advanced cybersecurity model within UAV operations.

We organize the paper as follows: Section II researches the related work in the field. Section III briefly describes the capabilities of an adversary and the attacking scenario. Section IV provides the detailed steps of constructing Areo-LLM. Section VI presents the experimental results. Section VII concludes the paper and suggests our future directions.

II. Related Work

This section briefly reviews the related work in three research domains: 1) LLM application in IoT and embedded systems; 2) Smaller-scale LLM cohorts application; and 3) The application of SFT and RLHF.

The application of large language models (LLMs) in the Internet of Things (IoT) and embedded systems has gained significant attention in recent years. Qiu et al. [11] proposed EdgeFormer, an edge-based transformer model for on-device natural language processing tasks in IoT environments. Their work demonstrated the feasibility of deploying LLMs on resource-constrained edge devices. Similarly, Zhang et al. [12] introduced a deflating technique to compress pre-trained LLMs for efficient deployment on embedded systems while maintaining performance.

Several works have explored the use of smaller-scale LLM cohorts for specific tasks. Su et al. [13] proposed GlobalPipeline, a framework that decomposes large LLMs into smaller experts and orchestrates their collaboration. Their approach showed improved efficiency and scalability compared to monolithic LLMs. Likewise, Dai et al. [14] introduced a knowledge distillation method to train smaller LLMs from larger ones, enabling efficient deployment on edge devices.

Fine-tuning pre-trained LLMs has proven effective for adapting them to specific tasks and domains. Supervised fine-tuning (SFT) has been widely used to fine-tune LLMs on labeled data [15], [16]. Reinforcement learning from human feedback (RLHF) has also been explored as a fine-tuning approach, where human feedback is used to refine the LLM's behavior [17], [18].

III. Threat Model

In this section, we describe the potential attack vectors that can be detected by the LLM-empowered sub-system and the computing requirements for deploying LLMs at various levels of the system architecture.

Areo-LLM framework aims to detect and forecast the following attack vectors: 1) *network attacks* that attempt to disrupt these channels through jamming, spoofing, or man-in-the-middle attacks; 2) *sensor manipulation attacks* that manipulate sensor data or inject false information to mislead the UAV's decision-making processes; 3) *software vulnerabilities* that can be exploited by adversaries to gain unauthorized access or control; and 4) *insider threats* that attempt to disrupt operations or steal sensitive data.

A reliable and high-bandwidth network infrastructure is essential to enable real-time communication and data exchange between the UAV, edge servers, and cloud servers. For smaller LLMs deployed onboard the UAV, embedded GPUs or specialized AI accelerators with limited computational resources may be sufficient. At the edge and cloud level, more powerful GPU resources may be available, enabling the deployment of larger-scale LLMs for computationally intensive tasks, such as finetuning or inference with large context windows. By deploying LLMs at various levels of the system architecture, the Areo-LLM framework provides a robust and effective defense against various cyber threats targeting UAV systems.

IV. System Design

Figure 1 illustrates the sequential phases of data collection, LLM fine-tuning, and deployment for UAV systems. The UAV data was collected from two sources: the digital twins and the actual UAVs connected with software-in-the-loop (SITL) and hardware-in-the-loop (HITL) environments, using tools, such as PixelHawk and ArduPilot to simulate and capture data, which is then fed into a central repository. This repository collects both simulated and real-world sensor data, vital for fine-tuning task-specific LLMs. The fine-tuning of LLMs occurs on the cloud server, using the training data to ensure the models are well-adapted to the operational context of UAVs. Once fine-tuned, these models are deployed on edge computing platforms, leveraging the edge server equipped with GPUs. This enables efficient and rapid processing of UAV data in real-time, optimizing performance and responsiveness for mission-critical applications.

A. LLM Fine-tuning and Deployment

Figure 2 illustrates the interaction between different components, including the Ground Control Station (GCS), the UAV, the cloud, and edge servers. The cloud server performs offline processes, including fine-tuning the LLM for enhanced inferencing, ensuring the LLM's responsiveness is precisely calibrated for anomaly detection.

During active missions, the UAV transmits real-time sensor data to the edge server for immediate processing—an online process leveraging edge computing to minimize latency. The edge Server employs the LLM to analyze incoming data streams in real time, ensuring swift anomaly detection. Critical performance metrics such as accuracy, precision, recall, and

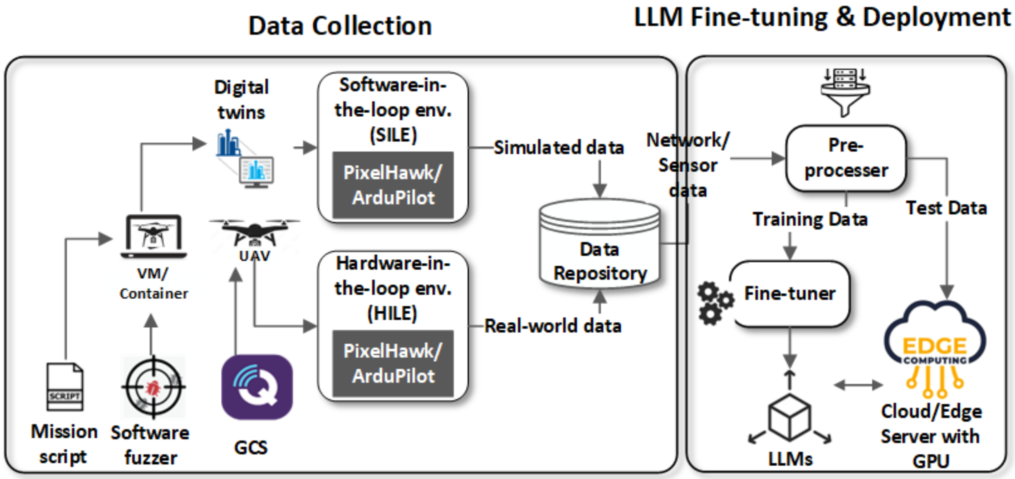


Fig. 1: System Architecture of Areo-LLM

F1-Score are continuously monitored to maintain operational integrity.

When anomalies are detected, the system generates detailed anomaly reports. These reports serve as actionable insights for the GCS, informing potential mission adjustments and contributing to the iterative enhancement of the UAV's operational framework. This closed-loop system exemplifies the integration of edge AI and real-time analytics in UAV operations, prioritizing immediate data processing and adaptive learning for continual improvement of mission-critical tasks.

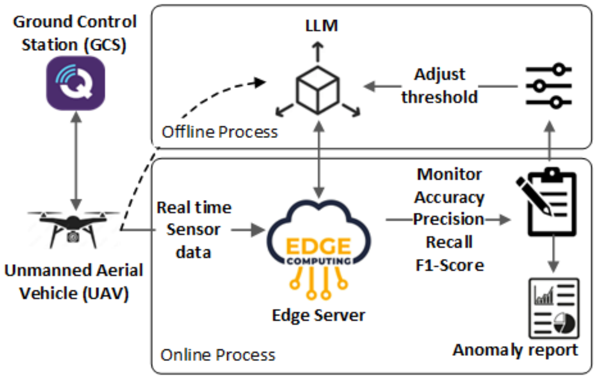


Fig. 2: LLM Fine-tuning and Deployment

V. Implementation

A. Preprocessing Data

1) Network layer data: We take the following steps to process network packets. First, we single out TCP sessions from our System-in-the-Loop (SIL) simulation. Then, we convert the sessions into a sliding window dataset. Using the format shown in Figure 4, we use one message as the question, along with previous messages serving as the context. Following this, we have packets labeled as 'chosen' and 'rejected.' The 'chosen' packet is the ground truth, representing the following packet in the session, while the 'rejected' packet is identical to the 'chosen' one except for one of the six key-value pairs being changed to an incorrect value. We collect TCP sessions, UDP sessions, and MavLink sensor data for normal and anomalous missions by using Software in the Loop (SIL).

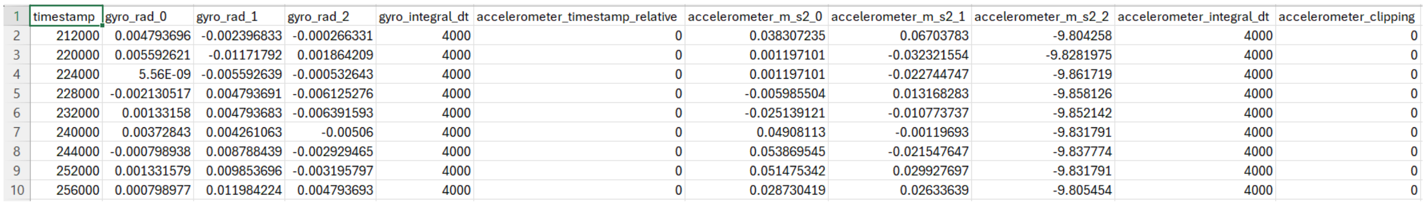
We compose the fine-tuning data in the format as illustrated in Fig. 4. Each data entry includes three sections: `#Previous_Packet`, `#Predicted_Packet`, and `#Context`, which use the same `#BLOCK` section that specifies `<key:value>` pairs.

2) Application layer data: We collect the sensor data from uLog generated from the missions as described in Section V-B2. The uLog data is parsed into CSV file using PX4 log utils. The CSV files are used for fine-tuning the models Time-LLM [19] and TimesNet [20]. We chose combined sensor data for the experiments that contain sensor data from the Gyro and Accelerometer. The format of the data is shown in Figure 3.

B. Fine-tuning LLMs

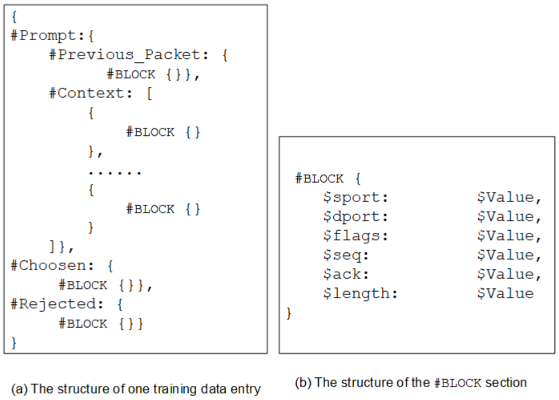
1) TCP/UDP Data: The fine-tuning process utilizes a three-stage approach introduced by Microsoft's DeepSpeed [21]. The initial stage involves training an actor model through Supervised Fine-Tuning (SFT), a method where the model learns from a labeled dataset to predict accurate outputs. In the second stage, a critic model is trained through a human feedback loop, enabling the model to judge the accuracy of each value in a predicted packet with precision. The final stage, and arguably the most critical, involves Iterative Reinforcement Learning from Human Feedback (RLHF) Refinement. This stage deploys the RLHF model to further fine-tune the SFT model, resulting in a significantly enhanced final model that benefits from precise supervision and iterative feedback-driven adjustments.

2) MavLink Sensor data: PX4 sensor data is a time series. Various sensors in UAVs communicate by sending messages on various topics. We collect data from the Ulog and convert



1	timestamp	gyro_rad_0	gyro_rad_1	gyro_rad_2	gyro_integral_dt	accelerometer_timestamp_relative	accelerometer_m_s2_0	accelerometer_m_s2_1	accelerometer_m_s2_2	accelerometer_integral_dt	accelerometer_clipping
2	212000	0.004793696	-0.002396833	-0.000266331	4000	0	0.038307235	0.06703783	-9.804258	4000	0
3	220000	0.005592621	-0.01171792	0.001864209	4000	0	0.001197101	-0.032321554	-9.8281975	4000	0
4	224000	5.56E-09	-0.005592639	-0.000532643	4000	0	0.001197101	-0.022744747	-9.861719	4000	0
5	228000	-0.002130517	0.004793691	-0.006125276	4000	0	-0.005985504	0.013168283	-9.858126	4000	0
6	232000	0.00133158	0.004793683	-0.006391593	4000	0	-0.025139121	-0.010773737	-9.852142	4000	0
7	240000	0.00372843	0.004261063	-0.00506	4000	0	0.04908113	-0.00119693	-9.831791	4000	0
8	244000	-0.000798938	0.008788439	-0.002929465	4000	0	0.053869545	-0.021547647	-9.837774	4000	0
9	252000	0.001331579	0.009853696	-0.003195797	4000	0	0.051475342	0.029927697	-9.831791	4000	0
10	256000	0.000798977	0.011984224	0.004793693	4000	0	0.028730419	0.02633639	-9.805454	4000	0

Fig. 3: The Format of Sensor Data



```
{
#Prompt:{
    #Previous_Packet: {
          #BLOCK {}},
    #Context: [
        {
            #BLOCK {}
        },
        ......
        {
            #BLOCK {}
        }
    ]},
#Choosen: {
     #BLOCK {}},
#Rejected: {
     #BLOCK {}}
}
```

(a) The structure of one training data entry

```
#BLOCK {
   $sport:        $Value,
   $dport:        $Value,
   $flags:        $Value,
   $seq:          $Value,
   $ack:          $Value,
   $length:       $Value
}
```

(b) The structure of the #BLOCK section

Fig. 4: The Format of Data Under the fine-tuning.

TABLE I: Accuracy of Predicted Fields (in %)

Field	SFT & RLHF
OPT-350	
sport	100.00
dport	100.00
flags	99.85
seq	98.45
ack	48.10
length	99.95
0 errors: 53.23, 1 error: 45.29,	2 errors: 1.11; 3 errors: 0.37, 4+ errors: 0
OPT-1.3B	
sport	100.00
dport	100.00
flags	99.45
seq	98.34
ack	53.97
length	99.63
0 errors: 47.30, 1 error: 51.80,	2 errors: 0.85; 3 errors: 0.05, 4+ errors: 0

it into CSV format, which will be input into the model for fine-tuning.

We select two models for different tasks: TimesNet for anomaly detection and Time-LLM for forecasting. We first gather historical data from UAV sensors to fine-tune the Time-LLM for forecasting. This data undergoes preprocessing, including filling in missing values, normalization, and division into training, validation, and test subsets. It's then tokenized to match Time-LLM's language, trained on the training set, and evaluated on the test set to gauge its forecasting performance. Accuracy and precision are measured by calculating MAE and MSE. Finally, the optimized model is saved for future predictions.

To finetune the TimesNet model for anomaly detection, we commence by collecting UAV sensor data, such as altitude and GPS coordinates, followed by preprocessing that includes imputation of missing values and feature scaling. We divide the dataset into subsets, ensuring a mix of normal and anomalous points, and label them based on historical anomaly records. With the TimesNet model initialized with pre-trained weights, we then set the fine-tuning parameters, such as sequence length and learning rate. After training on the dataset until performance plateaus, the refined model is saved and ready for real-time anomaly detection in UAV operations.

C. *Cyber Security tasks*

We choose the TimesNet [20] model for anomaly detection and train it using the preprocessed PX4 Sensor data.After the model is fine-tuned using PX4 Sensor data, the anomaly detection script is run using the test data set with labels.

We choose Time-LLM [19] model based on the Llama-2-7B to finetune the model for forecasting. Time-LLM reprograms large language models for time series without altering the pre-trained foundation model.Time-LLM model performs better than state-of-the-art solutions in few-shot and zero-shot scenarios.

Algorithm 1 Anomaly Detection using TimesNet

```
Inputs:
    $\mathcal{M}$: TimesNet Model,
    $\Theta$: Fine-tuned Model Weights and Biases,
    $\mathcal{D}$: Real-time Sensor Data,
    $E$: Number of Epochs,
    $B$: Batch Size
Output:
    $\alpha$: Accuracy, $\pi$: Precision, $\rho$: Recall, $\phi$: F-Score
Initialize:
    $\mathcal{P} \leftarrow \emptyset$; $\mathcal{G} \leftarrow \emptyset$
    $\tau \leftarrow predefined\_value$; $e \leftarrow 1$
while $e \leq E$ do
    for each $batch \in \mathcal{D}$ do
        $x, y \leftarrow batch$
        $p \leftarrow \mathcal{M}.predict(x)$
        Append $p$ to $\mathcal{P}$
        Append $y$ to $\mathcal{G}$
    $e \leftarrow e + 1$
$\mathcal{A} \leftarrow detect\_anomalies(\mathcal{P}, \mathcal{G}, \tau)$
Calculate evaluation metrics:
$\alpha \leftarrow calculate\_accuracy(\mathcal{P}, \mathcal{G})$
$\pi \leftarrow calculate\_precision(\mathcal{P}, \mathcal{G})$
$\rho \leftarrow calculate\_recall(\mathcal{P}, \mathcal{G})$
$\phi \leftarrow calculate\_FScore(\mathcal{P}, \mathcal{G})$
Return $\alpha, \pi, \rho, \phi$
```

We set the following parameters for fine-tuning TimesNet and Time-LLM models :

TABLE II: Parameters for finetuning Large models

Training parameters	Model	
Parameter	TimesNet	Times-LLM
Epochs	3	1
Learning Rate	0.02	0.02
Llama layers	N/A	8
Batch Size	128	4
Model Dimension	64	32
FCN Dimension	64	128

The metrics from fine-tuning:

TABLE III: Metrics From TimesNet Finetuning

Metrics	
Train Loss	0.1713483
Validation Loss	0.1251146
Test Loss	0.1068001
MAE Loss	0.2587003
Learn Rate	0.0008000000

VI. Evaluation

1) Anomaly Detection: To enrich our dataset with anomalous data, we employ several techniques to transform normal records. Firstly, we designate every n-th record as anomalous. Additionally, we introduce irregularities by randomly altering select records. To simulate different degrees of deviation, we systematically vary the dataset's variance and generate corresponding datasets. Lastly, we apply a Poisson distribution to intersperse anomalous data throughout the dataset, ensuring a realistic distribution of anomalies for robust model training.

The following formulas are used for detecting anomalies. The loss is calculated as:

$$loss = MSE(predicted\_value - groundtruth\_value) \quad (1)$$

The anomaly threshold is calculated as:

$$threshold = Percentile(loss, 100 - anomaly\_ratio) \quad (2)$$

a) Every nth record is anomaly: We take the normal data from the sensor and manipulate every nth record to contain anomalous data. For the experiment, we choose n=5. The anomalous data thus generated is fed into the TimesNet model running on the Edge Server, and the loss is calculated as per formula 1. When the loss exceeds the thresholds as calculated in formula 2, the data is flagged as an anomaly. The anomaly metrics are illustrated in Figure 6

b) Change variance of anomalous data: Sensor reading accelerometer_m_s2_2 is arbitrarily changed to simulate variances and the anomaly detection script is run for each variance and the data is collected. The metrics from changing the variance is calculated and plotted in Figure The anomaly metrics are illustrated in Figure 6.

From the test results and from the graph in figure 7b, we can infer that the TimesNet model is very resilient, and the variance in data doesn't seem to impact a lot on the metrics such as accuracy, precision, recall, and F-Score.

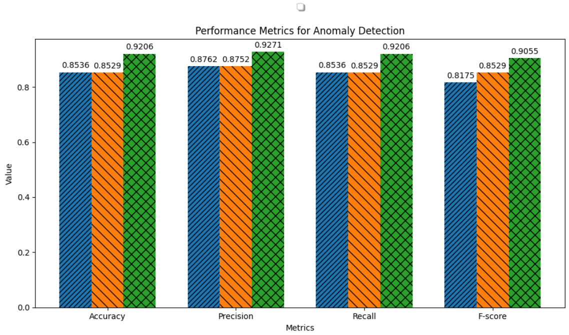


Fig. 5: Metrics From TimesNet Anomaly Detection

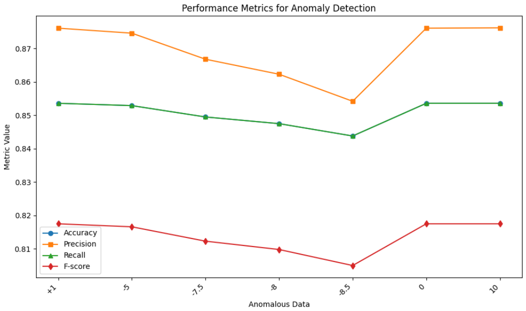


Fig. 6: Anomalous Data Vs Metrics

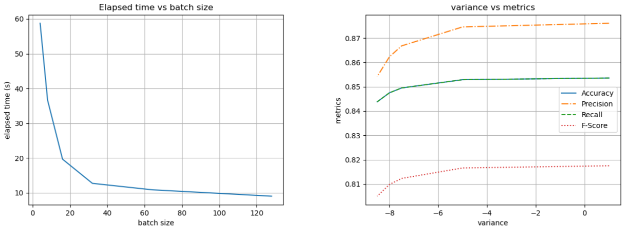
c) Impact of batch size: We evaluate the relationship between the batch size and time to detect anomalies. We set one sensor reading to an arbitrary value of -8.5 and run the anomaly detection script by just changing the batch size and keeping all other parameters. All the runs produce the same metrics as Accuracy : 0.8438, Precision : 0.8542, Recall : 0.8438, F-score : 0.8050. From the table IV of experiment results and from the graph 7a, we can infer that the batch size plays a significant role in the sensitivity of the anomaly detection. Batch size 48 and above significantly reduced the elapsed time to detect anomalies. Also, increasing the batch size beyond 48 doesn't reduce the elapsed time proportionally.

TABLE IV: Metrics From TimesNet for various batch size

Metrics - Batch size	
Batch Size	Avg Elapsed Time
128	9.08162
64	10.89548
32	12.71874
16	18.21896
8	34.2963
4	61.91978

Please note that the accuracy and recall lines are not distinguishable as the accuracy and recall data are the same. So, the lines appear in Green color.

2) Forecasting: We run the forecasting script that uses a finetuned Time-LLM model for the test dataset, and We record the loss for the test data. The test data is taken from one mission and consists of normal data. The model is able to



(a) Batch vs elapsed time (b) Variance vs Metrics

Fig. 7: Comparative analysis of TimesNet performance

predict the future data accurately. We calculate the test Loss is equal to *0.1068001* and MAE Loss is equal *0.2587003*

The results show the accuracy is > 82%.

VII. Conclusion

In this paper, we introduce Areo-LLM framework in the field of UAV operations, addressing the critical need for secure, efficient, and intelligent systems. By integrating specialized Large Language Models (LLMs) within a collaborative framework, Areo-LLM enhances the capabilities of UAVs in performing complex missions, ensuring high levels of security and operational efficiency. Our comprehensive evaluation of Areo-LLM has confirmed its high performance in key metrics such as accuracy, precision, recall, and F1 score, while maintaining a low memory footprint.

VIII. Acknowledgment

This work was supported in part by the National Aeronautics and Space Administration (NASA) award 80NSSC20M0124 and Oakland University URC faculty research fellowship.

References

[1] Z. Wang, L. Zhang, and P. Liu, “ChatGPT for Software Security: Exploring the Strengths and Limitations of ChatGPT in the Security Applications,” 2023.

[2] E. N. Crothers, N. Japkowicz, and H. L. Viktor, “Machine-generated text: A comprehensive survey of threat models and detection methods,” *IEEE Access*, vol. 11, 2023.

[3] H. Pearce, B. Tan, P. Krishnamurthy, F. Khorrami, R. Karri, and B. Dolan-Gavitt, “Pop quiz! can a large language model help with reverse engineering?” *CoRR*, vol. abs/2202.01142, 2022.

[4] N. Jain, S. Vaidyanath, A. Iyer, N. Natarajan, S. Parthasarathy, S. Rajamani, and R. Sharma, “Jigsaw: Large language models meet program synthesis,” in *Proceedings of the 44th International Conference on Software Engineering*, ser. ICSE ’22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1219–1231. [Online]. Available: https://doi-org.huaryu.kl.oakland.edu/10.1145/3510003.3510203

[5] M. A. Ferrag, M. Ndhlovu, N. Tihanyi, L. C. Cordeiro, M. Debbah, and T. Lestable, “Revolutionizing cyber threat detection with large language models,” 2023.

[6] G. Sandoval, H. Pearce, T. Nys, R. Karri, S. Garg, and B. Dolan-Gavitt, “Lost at c: A user study on the security implications of large language model code assistants,” 2023.

[7] Meta AI, “Llama 2: Enhancing large language models,” Accessed: March 28, 2024. [Online]. Available: https://huggingface.co/blog/llama2

[8] Google DeepMind, “Gemini: A family of highly capable multimodal models,” Accessed: March 28, 2024. [Online]. Available: https://huggingface.co/papers/2312.11805

[9] Mistral AI, “Mistral: Generating training data for large language models,” Accessed: March 28, 2024.

[10] Databricks, “Dbrx base, a mixture-of-experts (moe) large language model,” Accessed: March 28, 2024. [Online]. Available: https://huggingface.co/databricks/dbrx-base

[11] H. Qiu, J. Qiu, Y. Li, Q. Li, and J. Li, “Edgeformer: Transformer-based natural language processing on edge devices,” *arXiv preprint arXiv:2205.12487*, 2022.

[12] J. Zhang, Z. Li, Y. Xue, C. Zhang, and D. Li, “Deflating pre-trained language models for efficient deployment on embedded systems,” *arXiv preprint arXiv:2212.08015*, 2022.

[13] Z. Su, Y. T. Tan, Y. Liang, H. Choi, H. Park, S. Chen, X. Lin *et al.*, “Globalpipeline: An efficient model parallelism via simple data-parallelism principles,” in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

[14] D. Dai, C. Li, and B. Peng, “Knowledge distillation for small-footprint efficient transformers,” in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022.

[15] T. Gao, A. Fisch, and D. Chen, “Making pre-trained language models better few-shot learners,” *arXiv preprint arXiv:2103.03511*, 2021.

[16] K. Lee, D. Ippolito, E. Nouri, R. L. Bras, C. Callison-Burch, and J. Sedoc, “Deduplicating training data makes language models better,” *arXiv preprint arXiv:2203.06642*, 2022.

[17] N. Stiennon, L. Ouyang, J. Ziegler, R. Byrne, A. Radford, L. Paull, A. Bachand, D. Kim, P. M. D. Moore *et al.*, “Learning to summarize from human feedback,” *arXiv preprint arXiv:2009.01325*, 2020.

[18] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, “Training language models to follow instructions with human feedback,” *arXiv preprint arXiv:2203.02155*, 2022.

[19] M. Jin, S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan, and Q. Wen, “Time-LLM: Time series forecasting by reprogramming large language models,” in *International Conference on Learning Representations (ICLR)*, 2024.

[20] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, “Timesnet: Temporal 2d-variation modeling for general time series analysis,” in *International Conference on Learning Representations*, 2023.

[21] R. Yazdani Aminabadi, S. Rajbhandari, M. Zhang, A. A. Awan, C. Li, D. Li, E. Zheng, J. Rasley, S. Smith, O. Ruwase, and Y. He, “Deepspeed inference: Enabling efficient inference of transformer models at unprecedented scale,” *arXiv preprint arXiv:2207.00032*, 2022.